THE DYNAMIC CHARACTERISTICS OF COMPUTER PROGRAMS

G. E. Bryan

August 1967

P-3661

28

THE DYNAMIC CHARACTERISTICS OF COMPUTER PROGRAMS

G. E. Bryan*

The RAND Corporation, Santa Monica, California

One of the goals continuously before us in computer programming is to seek out ways to improve, by shortening, the amount of time taken to solve problems on digital computers. One way to achieve this goal is through more efficient use of the computing facilities which we have today. Monitor systems made a great stride toward this goal by automating the sequencing of jobs through the computing machine, making available on call a number of helpful programs for compiling, assembling, converting, and editing. For a number of years a great white hope has been the multiplexing of programs within a single computing machine. It is hoped that through this technique, idle times of one program may be interleaved with computing times of a second or third program with the combination making more efficient use of the machine than any one of them could have made by itself.

---

*Any views expressed in this Paper are those of the author. They should not be interpreted as reflecting the views of The RAND Corporation or the official opinion or policy of any of its governmental or private research sponsors. Papers are reproduced by The RAND Corporation as a courtesy to members of its staff.

This Paper was presented at the Twenty-Second Annual Meeting of SHARE, held in San Francisco, 2-6 March 1964.

In multiplex program situations, important questions arise regarding storage allocation, algorithms for choosing the jobs to be run, proper timing of swapping of programs in and out of main memory, and algorithms for switching between programs which are already in memory. In order to answer these questions and to design efficient algorithms, a number of questions about the characteristics of the programs being run need to be answered. Algorithms which operate effectively in one mix of program characteristics will operate inefficiently in other mixes. Some of the central questions involved are:

1) The distribution of program sizes.

2) The distribution of running times of programs.

3) The correlation, if any, between 1 and 2.

4) The characteristics of the use of storage by programs.

5) Identification of idle intervals within a program, and finding their time distribution.

In category 5, two areas seem apparent. The first is stop time between jobs for tape mounting, finding the next job, dumping the program, etc.; the second, those intervals during which a program waits for I/O actions to complete.

Determining the answers to these questions in current operations will help us know whether program multiplexing is indeed a fruitful area for efficiency gains, and, if it is, help us to find out what kinds of gains we can expect. We should not, of course, neglect the fact that the

job characteristics which we measure today are in some degree dependent on the current method of operation. Programmers, quite correctly, adapt their methods of operation to the system in which they work; any change in the system can be expected to change their habits somewhat. It is entirely possible, of course, that a system design for multiplex program operation will find greater utility and greater solution efficiency than current systems because of the easing of the difficulties of storage allocation and secondary storage utilization.

In order to examine some of these questions in detail, and to find the characteristics of programs using the computer at RAND, a number of statistics have been gathered on actual program runs. Preliminary results of these studies are given below.

Storage Examination Program

The FORTRAN EXIT program, entered at the end of nearly every FORTRAN job, has been modified to include a routine which examines storage following the execution of each job. Three categories of jobs are not reflected in the statistics gathered by this program:

1) Jobs which compile only.

2) Jobs so large that they cannot afford storage for this program.

3) Jobs which are dumped from the machine before reaching the EXIT routine.*

The storage scan routine breaks core into three areas: the program area, below the program break; the common area, above the common break; and unused core, that portion between the program break and the common break. Within each of these areas, five types of cells are recorded:

1) Those which are classed as decrement integers. If prefix tag and address of the word are 0, then it is assumed to be a decrement integer.

2) Zeros, both plus and minus.

3) Instructions which have nine bits equal to one (AXT, TSX, LXD, SXD, etc.).

4) Floating-point numbers--those which have the nine bit equal to one, except for those falling in category 3.

5) All other cells which are presumably instructions.

The program punches an accounting card containing these data, and this accounting card is later combined with the ordinary accounting cards produced for the run. Specific additional data gained by combining with the regular accounting cards are log on time, execution time, and number of output lines produced by the program.

Finally, the results of these cards for each job are summarized by a program which produces histograms of the data in various categories.

---

*The limitation of this last category has been eliminated as of 1 February 1964 by including the storage scan routine in the dump routine.

Preliminary Results

Data has been gathered with this storage examination program since 30 December 1963, and histograms have been run on those programs run between 30 December 1963 and 24 January 1964. About 150 jobs were run each day, two-thirds of them in prime shift.

The gross breakdowns of all jobs are shown in Fig. 1. The histograms described below are taken from statistics cards produced by the 60 percent of programs marked in Fig. 1 "FORTRAN Compile and Execute." As can be seen, these jobs represent 48 percent of the running time. The jobs were run on 19 different days during the period.

Figure 2 is the distribution of the number of 0's found in the region of core between the program break and common; thus, it is the best measure we know of the number of cells not used by the program. The bucket, or interval size is given under the heading BKT, with the actual count of jobs and the percentage of jobs falling in each bucket being given in the first two columns. Thus, the figure shows that 64, or 3.7 percent of jobs, did not use between 0 and 1000 cells of core. In this particular bucket, representing very large jobs, the true figure would be larger by perhaps as much as 10 percent since a substantial fraction of jobs are not reflected because they were too large to use this special exit program.

Since the normal complement of FORTRAN library routines occupies approximately 2500 to 3000 cells of memory, it is

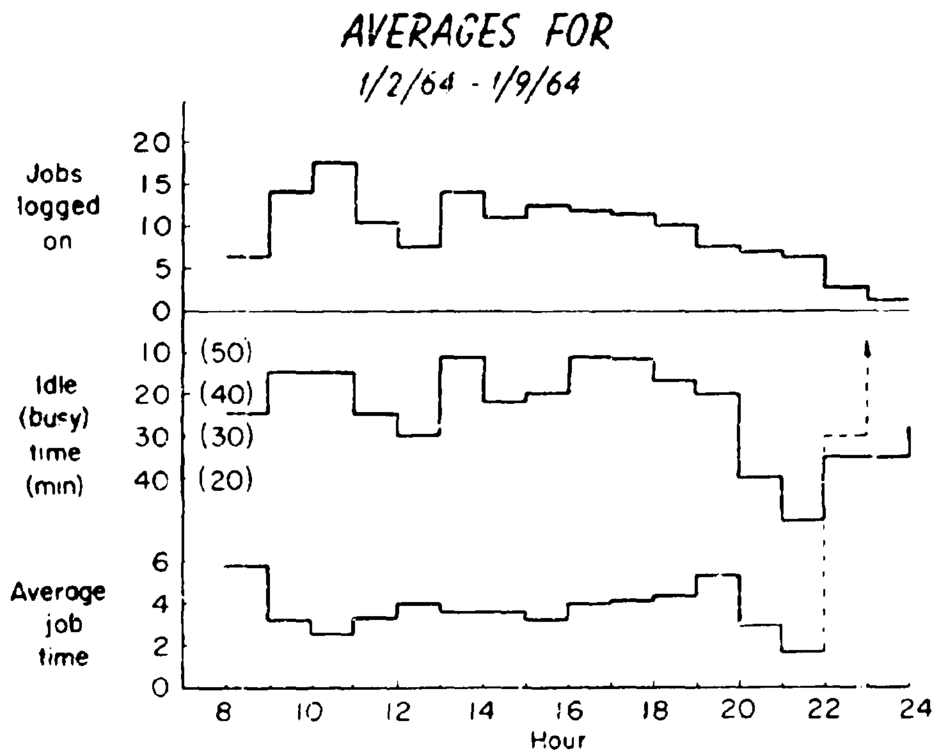
AVERAGES FOR
1/2/64 - 1/9/64
Jobs logged on
20
15
10
5
0
Idle (busy) time (min)
10 (50)
20 (40)
30 (30)
40 (20)
Average job time
6
4
2
0
8
10
12
14
16
18
20
22
24
Hour

FIGURE 1

```
TOTALS                              UNUSED ZEROS    (NOT USED, NOT REQUESTED CORE)

COUNT PRCNT BKT                                            HISTOGRAM

   64   3.7  0K XXXXXXX
   54   3.1  1K XXXXXX
   34   2.0  2K XXXX
   24   1.4  3K XXX
   32   1.8  4K XXXX
   78   4.5  5K XXXXXXXXX
   34   2.0  6K XXXX
   26   1.5  7K XXX
   47   2.7  8K XXXXX
   84   4.8  9K XXXXXXXXXX
   34   2.0 10K XXXX
   55   3.2 11K XXXXXX
   93   5.4 12K XXXXXXXXXXX
   57   3.3 13K XXXXXXX
   35   2.0 14K XXXX
   75   4.3 15K XXXXXXXXX
   38   2.2 16K XXXX
   29   1.7 17K XXX
   18   1.0 18K XX
   18   1.0 19K XX
   28   1.6 20K XXX
   44   2.5 21K XXXXX
   36   2.1 22K XXXX
   72   4.2 23K XXXXXXXX
   67   3.9 24K XXXXXXXX
   72   4.2 25K XXXXXXXX
   64   3.7 26K XXXXXXX
   52   3.0 27K XXXXXX
  175  10.1 28K XXXXXXXXXXXXXXXXXXXX
  169   9.8 29K XXXXXXXXXXXXXXXXXXXX
   24   1.4 30K XXX
    0    .0 31K
    0    .0 32K

 1732     TOTAL
```

FIGURE 2

not surprising that few jobs leave more than 30,000 cells unused. It is interesting, however, that 36 percent of all programs would have run successfully in an 8K machine.

Two programs, ROCKET and SIMSCRIPT, seem to account for the bulge in storage use in the 12-15K buckets. We should not find it surprising that our machine utilization characteristics are affected by popular programs.

Interestingly, only about 8 percent of programs would not fit in core with some other program commonly available.

Figure 3 presents the number of 0's found anywhere in memory--program area, unused storage, and common. Interesting is the fact that 90 percent of programs leave half of memory or more zero following their execution; 30 percent of programs leave 90 percent of memory empty. Examination of Fig. 3--and Fig. 4 (which shows that the number of floating-point cells found in memory is surprisingly small--70 percent of programs with less than 2000 floating-point numbers)--makes immediately apparent that many programs contain sparse matrices or allocated but unused tables. Clearly, in the case of sparse matrices, storage allocation techniques of the list variety such as IPL-V possesses would be of great value; and, in the second case, a dynamic storage allocation scheme providing storage only when requested would save large amounts of storage for use by other programs.

Figure 5 shows actual program sizes as determined by the number of instructions in the program and unused region.

```
TOTALS                                   TOTAL OF ZERO CELLS

COUNT  PRCNT  BKT                                                            HISTOGRAM

    0     .0   0K
    0     .0   1K
    0     .0   2K
    0     .0   3K
    0     .0   4K
    1     .1   5K
    1     .1   6K
    3     .2   7K
    2     .1   8K
    8     .5   9K X
    9     .5  10K X
    4     .2  11K
   14     .8  12K XX
   57    3.3  13K XXXXXXX
   28    1.6  14K XXX
   26    1.5  15K XXX
   44    2.5  16K XXXXX
   36    2.1  17K XXXX
   91    5.3  18K XXXXXXXXXXX
   59    3.4  19K XXXXXXX
   43    2.5  20K XXXXX
   70    4.0  21K XXXXXXXX
   78    4.5  22K XXXXXXXXX
   70    4.0  23K XX  XXXX
  108    6.2  24K XXXXXXXXXXXX
   95    5.5  25K XXXXXXXXXXX
   94    5.4  26K XXXXXXXXXXX
  144    8.3  27K XXXXXXXXXXXXXXXXX
  144    8.3  28K XXXXXXXXXXXXXXXXX
  382   22.1  29K XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  121    7.0  30K XXXXXXXXXXXXXX
    0     .0  31K
    0     .0  32K

 1732      TOTAL
```

FIGURE 3

TOTALS TOTAL OF FLOATING POINT CELLS ALL JOBS

COUNT	PRCNT	BKT	HISTOGRAM
917	52.5	0*	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
324	18.7	1*	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
167	9.6	2*	XXXXXXXXXXXXXXXXXXXXX
54	5.6	3*	XXXXXXXXXXXX
46	2.7	4*	XXXXXX
73	4.2	5*	XXXXXXXXX
29	1.7	6*	XXX
20	1.2	7*	XX
17	1.0	8*	XX
10	.6	9*	X
9	.5	10*	X
5	.3	11*	X
1	.1	12*	
2	.1	13*	
7	.4	14*	X
2	.1	15*	
4	.2	16*	
0	.0	17*	
0	.0	18*	
0	.0	19*	
0	.0	20*	
0	.0	21*	
0	.0	22*	
0	.0	23*	
0	.0	24*	
0	.0	25*	
0	.0	26*	
0	.0	27*	
0	.0	28*	
0	.0	29*	
0	.0	30*	
0	.0	31*	
0	.0	32*	
1732	TOTAL		

FIGURE 4

```
 TOTALS                                 TOTAL CF PRESUMEC INSTRUCTIONS, PRCG + UNLSED

COUNT PRCNT BKT                                                    HISTOGRAM

    0    .0   0K
   67   3.9   1K XXXXXXXX
  552  31.9   2K XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  256  14.8   3K XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  169   9.8   4K XXXXXXXXXXXXXXXXXXXX
  113   6.5   5K XXXXXXXXXXXXX
  160   9.2   6K XXXXXXXXXXXXXXXXXX
   85   4.9   7K XXXXXXXXXX
   55   3.2   8K XXXXXX
   55   3.2   9K XXXXXX
   47   2.7  10K XXXXX
   54   3.1  11K XXXXXX
   16    .9  12K XX
    1    .1  13K
    1    .1  14K
    8    .5  15K X
   69   4.0  16K XXXXXXXX
   20   1.2  17K XX
    1    .1  18K
    3    .2  19K
    0    .0  20K
    0    .0  21K
    0    .0  22K
    0    .0  23K
    0    .0  24K
    0    .0  25K
    0    .0  26K
    0    .0  27K
    0    .0  28K
    0    .0  29K
    0    .0  30K
    0    .0  31K
    0    .0  32K

 1732       TOTAL
```

FIGURE 5

With the exception of the popular SIMSCRIPT and ROCKET, which appear in the 16,000 instruction bucket, most programs are very small--50 percent being less than 4000 instructions long. This fact, coupled with the total allocation in Fig. 2, forces one to conclude that large amounts of storage are assigned to tables.

Figure 6 presents the data of Fig. 5 (weighted by straight multiplication) by the execution time of the program involved. Since very little change in the distribution is noted, we conclude that it is not possible to tell from the number of instructions in a program how long it will execute. I'm sure no one will be surprised by this fact.

Figure 7, however, plots the number of unused 0's weighted by execution time. This is the same data as in Fig. 6 and shows a very pronounced shift toward the larger programs near the top of the chart. The peak of Fig. 6 in the small program region is completely missing from Fig. 7. Thus, we note the strong correlation (as shown again in a different way below) between the size of the programs and their execution time. It seems to be a strong characteristic of programs that if the space requested is small, they will run a short time. If it is large, they will run a long time.

In the 5K and 8K buckets may be seen the pronounced effect of popular programs. This time, not particularly big ones but long running.

```
'TCTALS                                        TOTAL INSTRUCTICNS, WEIGHTED BY EXECUTE TIME

COUNT PRCNT BKT                                                       HISTOGRAM

    0     .C   0K
  182     .8   1K XX
 6933  28.9   2K XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 19C2   7.9   3K XXXXXXXXXXXXXXXX
 3625  15.1   4K XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 1154   4.8   5K XXXXXXXXXX
 2552  1C.6   6K XXXXXXXXXXXXXXXXXXXXX
 26C6  1C.9   7K XXXXXXXXXXXXXXXXXXXXXX
  259   1.1   8K XX
  796   3.3   9K XXXXXXX
  941   3.9  10K XXXXXXXX
 20C5   8.4  11K XXXXXXXXXXXXXXXXX
  330   1.4  12K XXX
    3    .0  13K
   25    .1  14K
  146    .6  15K X
  312   1.3  16K XXX
  161    .7  17K X
   19    .1  18K
   42    .2  19K
    C    .0  20K
    0    .0  21K
    0    .0  22K
    0    .0  23K
    0    .0  24K
    0    .0  25K
    C    .0  26K
    0    .0  27K
    0    .0  28K
    0    .0  29K
    0    .0  30K
    0    .0  31K
    0    .0  32K

24OC3       TOTAL
```

FIGURE 6

TOTALS

UNUSED ZEROS, WEIGHTED BY EXECUTE TIME

COUNT	PRCNT	BKT	HISTOGRAM
2038	8.5	0K	XXXXXXXXXXXXXXXXX
1720	7.2	1K	XXXXXXXXXXXXXX
978	4.1	2K	XXXXXXXX
343	1.4	3K	XXX
390	1.6	4K	XXX
2400	10.0	5K	XXXXXXXXXXXXXXXXXXXX
483	2.0	6K	XXXX
1027	4.3	7K	XXXXXXXXX
3630	15.1	8K	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
315	1.3	9K	XXX
188	.8	10K	XX
1193	5.0	11K	XXXXXXXXXX
813	3.4	12K	XXXXXXX
621	2.6	13K	XXXXX
463	1.9	14K	XXXX
689	2.9	15K	XXXXXX
516	2.1	16K	XXXX
374	1.6	17K	XXX
108	.4	18K	X
105	.4	19K	X
207	.9	20K	XX
483	2.0	21K	XXXX
555	2.3	22K	XXXXX
784	3.3	23K	XXXXXXX
1256	5.2	24K	XXXXXXXXXX
372	1.5	25K	XXX
145	.6	26K	X
644	2.7	27K	XXXXX
762	3.2	28K	XXXXXX
375	1.6	29K	XXX
26	.1	30K	
0	.0	31K	
0	.0	32K	
24003		TOTAL	

FIGURE 7

Figure 8 again demonstrates this shift by plotting the total number of 0's weighted by execution time. It, too, reinforces the thought that big programs run a long time and little programs run a short time. And remember, these are not instructions in the program but total allocation of storage. Thus, while we cannot tell how long a program might run by asking how many instructions the program contained, we can tell by asking how many instructions plus how many cells of tables does it contain.

Figure 9 plots the number of jobs arriving at the 7090 in each hour of the day. Here the bucket column represents an hour. Noteworthy are the first and second shift lunch-hour dips at 12:00 and 20:00 hours. Also apparent is the slow rise from early morning to full production at 10:00, and from lunch time until full production again at 2:00. It seems a full stomach is a bad thing for programmers. Perhaps we should hire only hungry programmers.

Production of output for printing is a crucial one in most installations, being one of the primary bottlenecks hindering fast turnaround time. Figure 10 plots the number of jobs occurring in each category of output volume. From the figure, it can be seen that more than 50 percent of jobs can be printed using a single 600-line-a-minute printer in a time equivalent to the running time of the job. Perhaps this is an indication that we should return to on-line printing. We are certainly okay if we have sufficient

TOTALS TCTAL ZERCS, WEIGHTEC BY EXECUTE TIME

COUNT	PRCNT	BKT	HISTOGRAM
0	.0	0K	
0	.0	1K	
0	.0	2K	
0	.0	3K	
0	.0	4K	
23	.1	5K	
26	.1	6K	
33	.1	7K	
29	.1	8K	
160	.7	9K	X
65	.3	10K	X
127	.5	11K	X
141	.6	12K	X
733	3.1	13K	XXXXXX
133	.6	14K	X
18C7	7.5	15K	XXXXXXXXXXXXXXX
2115	8.8	16K	XXXXXXXXXXXXXXXXXX
1278	5.3	17K	XXXXXXXXXXX
3456	14.6	18K	XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
1038	4.3	19K	XXXXXXXXX
770	3.2	20K	XXXXXX
1723	7.2	21K	XXXXXXXXXXXXXX
813	3.4	22K	XXXXXXX
1135	4.7	23K	XXXXXXXXX
1854	7.7	24K	XXXXXXXXXXXXXXX
1118	4.7	25K	XXXXXXXXX
770	3.2	26K	XXXXXX
1866	7.8	27K	XXXXXXXXXXXXXXXX
786	3.3	28K	XXXXXXX
1766	7.4	29K	XXXXXXXXXXXXXXX
198	.8	30K	XX
0	.0	31K	
0	.0	32K	
240C3		TOTAL	

FIGURE 8

```
TOTALS
                                    ONTIMES - EACH BUCKET IS AN HOUR

COUNT PRCNT BKT
                                                         HISTOGRAM
   17   1.0  0K XX
   16    .9  1K XX
    7    .4  2K X
    9    .5  3K X
    2    .1  4K
    0    .0  5K
    1    .1  6K
    0    .0  7K
   13    .8  8K XX
   80   4.6  9K XXXXXXXXX
  166   9.6 10K XXXXXXXXXXXXXXXXXXX
  152   8.8 11K XXXXXXXXXXXXXXXXXX
  107   6.2 12K XXXXXXXXXXXX
  126   7.3 13K XXXXXXXXXXXXXXX
  173  10.0 14K XXXXXXXXXXXXXXXXXXXX
  170   9.8 15K XXXXXXXXXXXXXXXXXXXX
  166   9.6 16K XXXXXXXXXXXXXXXXXXX
  139   8.0 17K XXXXXXXXXXXXXXXX
  113   6.5 18K XXXXXXXXXXXXX
  104   6.0 19K XXXXXXXXXXXX
   42   2.4 20K XXXXX
   65   3.8 21K XXXXXXXX
   36   2.1 22K XXXX
   28   1.6 23K XXX
    0    .0 24K
    0    .0 25K
    0    .0 26K
    0    .0 27K
    0    .0 28K
    0    .0 29K
    0    .0 30K
    0    .0 31K
    0    .0 32K

 1732       TOTAL
```

FIGURE 9

TOTALS EXECUTION LINES IN THOUSANDS OF LINES ALL JOBS

COUNT	PRCNT	BKT	HISTOGRAM
686	39.6	0K	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
168	9.7	1K	XXXXXXXXXXXXXXXXXX
132	7.6	2K	XXXXXXXXXXXXXXX
106	6.1	3K	XXXXXXXXXXXX
80	4.6	4K	XXXXXXXXXX
59	3.4	5K	XXXXXXX
71	4.1	6K	XXXXXXXX
43	2.5	7K	XXXXX
36	2.1	8K	XXXX
45	2.6	9K	XXXXX
34	2.0	10K	XXXX
52	3.0	11K	XXXXXX
23	1.3	12K	XXX
19	1.1	13K	XX
17	1.0	14K	XX
14	.8	15K	XX
13	.8	16K	XX
8	.5	17K	X
5	.3	18K	X
12	.7	19K	X
12	.7	20K	X
16	.9	21K	XX
20	1.2	22K	XX
9	.5	23K	X
14	.8	24K	XX
8	.5	25K	X
2	.1	26K	
4	.2	27K	
9	.5	28K	X
1	.1	29K	
8	.5	30K	X
10	.6	31K	X
0	.0	32K	
1732	TOTAL		

FIGURE 10

storage available with which to buffer the output so that bursts of printed output from the program can be smoothed for presentation to the printer.

Most installations have limits on the number of lines which may be produced during the critical prime shift hours; thus, those jobs which appear on this curve near the tail end--the high output jobs--can be expected to occur during the non-critical third shift.

Figure 11 weights 7090 on-time by the number of execution lines produced. Thus, it is a map of the printer load during the day. Compare this figure with Fig. 9 and note the attempt of the noon-time operator to run the jobs which produce more output, possibly the longer jobs, during his rather hectic session at the machine.

Running Times

Figure 12 presents a number of different distributions of running times of jobs taken from various studies. Although there is a large variation in the number of jobs run in any particular time category, it is clear that the great bulk of jobs run for only a short amount of time. The limitations usually imposed at computing installations make this no surprise. Still, it is a bit surprising to find 60 percent or more programs executed in less than two minutes.

Figure 13's scatter diagram presents a correlation of the execution-time data together with the total program-size

```
   TOTALS
                                             ENTIRE WEIGHTED BY EXEC LINES IN 1000'S OF LINES

 COUNT PRCNT BKT
                                                                      HISTOGRAM
   146    .6  0K X
   271   1.1  1K XX
    97    .4  2K X
    65    .3  3K X
    22    .1  4K
     0    .0  5K
    12    .0  6K
     0    .0  7K
    70    .3  8K X
  1073   4.4  9K XXXXXXXXXX
  1710   7.1 10K XXXXXXXXXXXXXXX
  1353   5.6 11K XXXXXXXXXXXX
  2067   8.5 12K XXXXXXXXXXXXXXXXXX
  1874   7.7 13K XXXXXXXXXXXXXXXX
  2768  11.4 14K XXXXXXXXXXXXXXXXXXXXXXXXX
  1567   6.5 15K XXXXXXXXXXXXXX
  1464   6.0 16K XXXXXXXXXXXXX
  2531  10.4 17K XXXXXXXXXXXXXXXXXXXXXXX
  1966   8.1 18K XXXXXXXXXXXXXXXXXX
  1749   7.2 19K XXXXXXXXXXXXXXXX
   735   3.0 20K XXXXXX
  1052   4.3 21K XXXXXXXXXX
   852   3.5 22K XXXXXXXX
   781   3.2 23K XXXXXX
     0    .0 24K
     0    .0 25K
     0    .0 26K
     0    .0 27K
     0    .0 28K
     0    .0 29K
     0    .0 30K
     0    .0 31K
     0    .0 32K

 24225       TOTAL
```

FIGURL 11

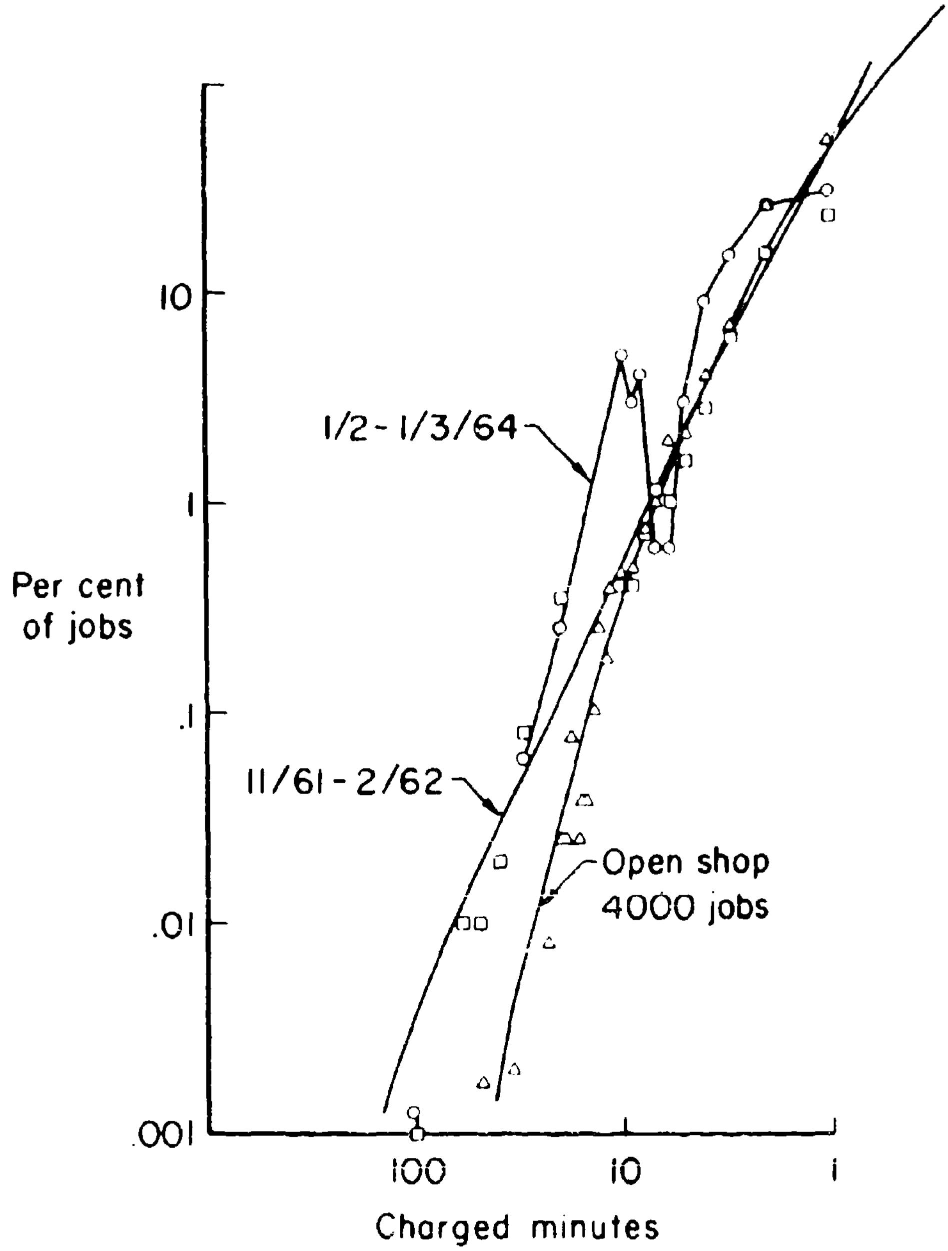
10
1
.1
.01
.001
Per cent
of jobs
1/2-1/3/64
11/61-2/62
Open shop
4000 jobs
100
10
1
Charged minutes

FIGURE 12

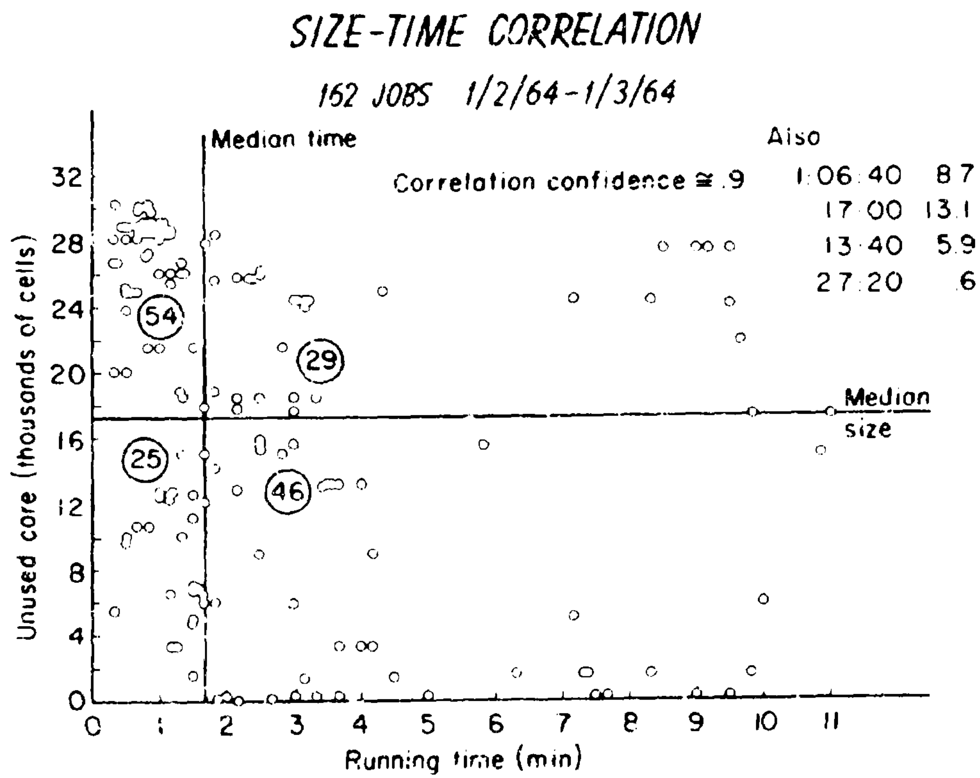
SIZE-TIME CORRELATION
162 JOBS 1/2/64-1/3/64
Median time
Also
Correlation confidence ≅ .9
1:06:40 8.7
17:00 13.1
13:40 5.9
27:20 6
Median size
54
29
25
46
Unused core (thousands of cells)
32
28
24
20
16
12
8
4
0
0 1 2 3 4 5 6 7 8 9 10 11
Running time (min)

FIGURE 13

data. As was seen in the histograms, there is a substantial correlation between job size and running time. Note particularly the large group of jobs in the under-one-minute running time and less than 4000-in-words of storage category. Again, since many of the jobs on the opposite end of the spectrum--the large end--are run late at night, we find small, short jobs a pronounced characteristic of prime shift jobs.

Unused Time

In our supposedly advanced monitor controlled job shops, we find substantial portions of unused time during even the busiest hours of the day. Figure 14 is an example of this characteristic. The center curve plots the average for 6 days of unused time during each hour. Note that we are never able to utilize more than 50 minutes out of each hour. Tape mounting, finding jobs, and other delays attributable to semi-manual operation are reflected here. It is probably unreasonable to expect that any non-automatic system would be able to do better.

Figure 14 also plots the number of jobs logged on each hour and the average time occupied by each job during that hour. Interestingly, between 10:00 in the morning and 8:00 in the evening the average job time does not vary substantially from the overall average time, although late at night large jobs were indeed run.

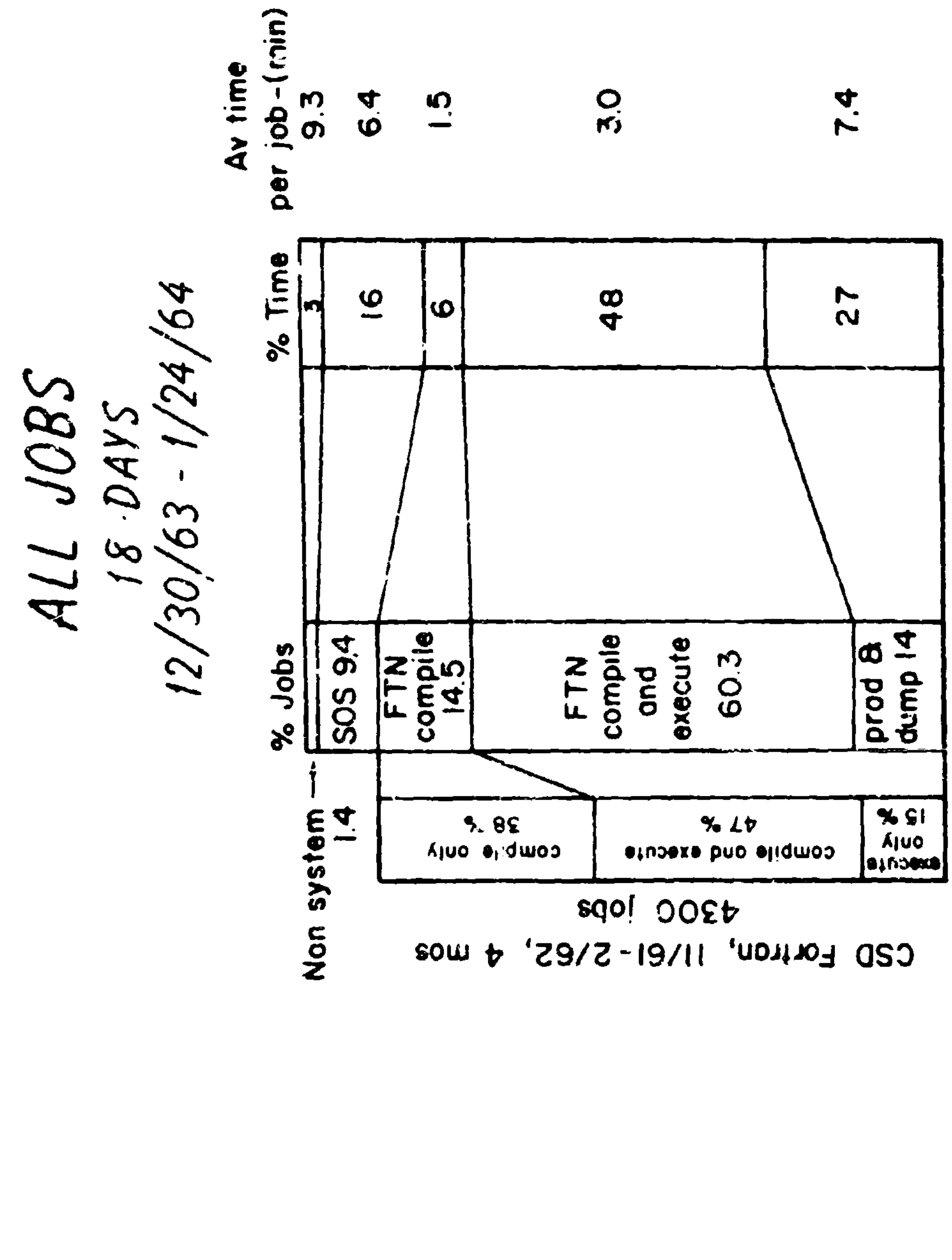
ALL JOBS
18 DAYS
12/30/63 - 1/24/64
% Jobs
SOS 9.4
FTN compile 14.5
FTN compile and execute 60.3
prod & dump 14
% Time
3
16
6
48
27
Av time per job-(min)
9.3
6.4
1.5
3.0
7.4
Non system — 1.4
compile only 38 %
compile and execute 47 %
execute only 15 %
4300 jobs
CSD Fortran, 11/61-2/62, 4 mos

FIGURE 14

It should be pointed out that the wait times reflected here are the non-charge times--and thus do not reflect waiting time within the program for tape I/0 transmission, rewinds, backspaces, and waits for the operator to dump the job, dial in tapes, or complete other manual operations.

Figure 15's scatter-diagram plots for each hour one point on coordinates of number of jobs logged in, and idle time for the hour. This diagram shows no correlation between idle time and number of jobs, indicating that the job-shop monitor is working fine but that certain manual delays are unavoidable and are independent of the number of jobs being processed.

Summary

I believe that the above statistics demonstrate that substantial gains are achievable through a multiplexed program mode of operation. Substantial numbers of programs exist in the small-time and small-program size categories to insure that programs can be easily found which will fit available time-space slots. Further, because of these factors, rather simple allocation algorithms will be sufficient. We need not look far ahead in the input stream to find a suitable job to fit the dimension available in either time or space.

We have also shown that there are substantial gains to be achieved in storage allocation areas--both in list processing styles of storage allocation in which both the

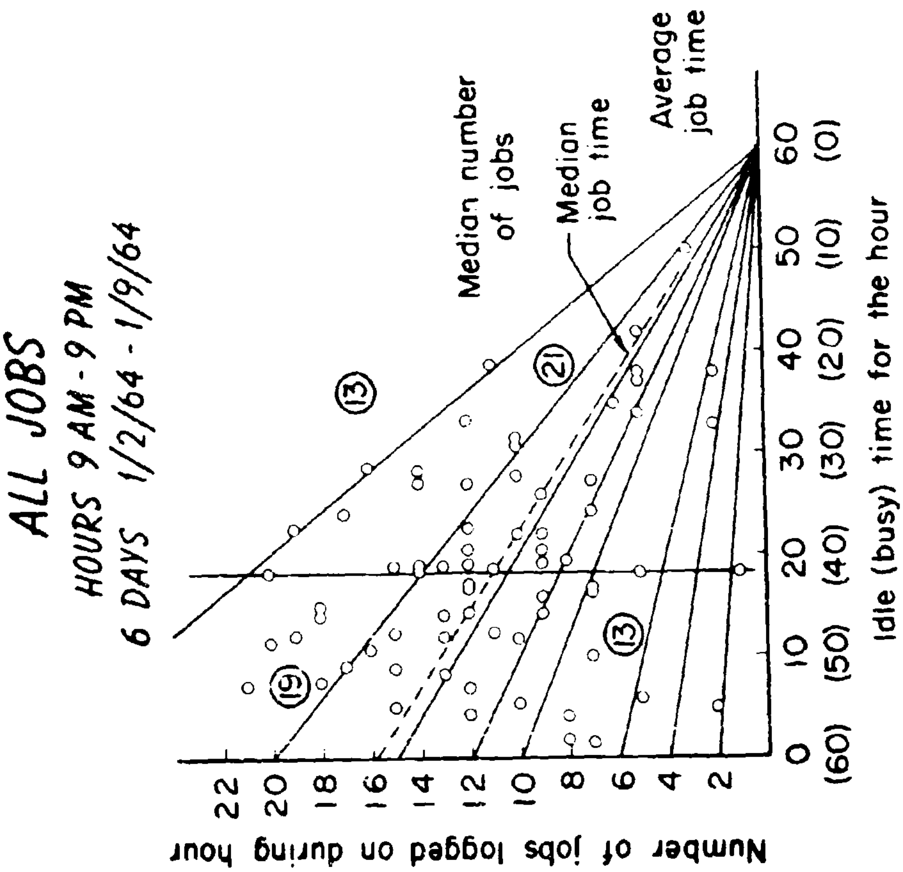
ALL JOBS
HOURS 9 AM - 9 PM
6 DAYS 1/2/64 - 1/9/64
Median number of jobs
Median job time
Average job time
13
21
19
13
Number of jobs logged on during hour
22
20
18
16
14
12
10
8
6
4
2
0
10
20
30
40
50
60
(60)
(50)
(40)
(30)
(20)
(10)
(0)
Idle (busy) time for the hour

FIGURE 15

location and its contents are important data in the storage reference, and in the dynamic storage requests in which tables of nominal size are expended to fill the needs of the program as it executes.

It would not be overstating the case to predict that an efficiency or through-put gain of 100 percent is achievable through implementation of these techniques.